Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Mojca Rojko

Fraktali

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Mentor: doc. dr. Žiga Virk

Ljubljana, 2015

Rezultati diplomskega dela so intelektualna lastnina avtorja. Za objavljanje ali izkoriščanje rezultatov diplomskega dela je potrebno pisno soglasje avtorja, Fakultete za računalništvo in informatiko ter mentorja.

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Fraktali

Tematika naloge:

Kandidatka naj obravnava tehnike in algoritme risanja fraktalov. Predstavi naj različne pristope, motivacije in matematična ozadja, ki vplivajo na fraktalno geometrijo objektov. Razvije naj aplikacijo, s katero bo moč generirati različne tipe fraktalov. Prav tako naj aplikacija uporabniku omogoča izbiro tipičnih matematičnih parametrov.

Izjava o avtorstvu diplomskega dela

Spodaj podpisana Mojca Rojko, z vpisno številko 63110265, sem avtor diplomskega dela z naslovom:

Fraktali

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom doc. dr. Žiga Virka,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 24. avgusta 2015 Podpis avtorja:

Zahvaljujem se mentorju doc. dr. Žigu Virku za ves vložen trud, ter za strokovno usmerjanje in svetovanje. Posebna zahvala gre tudi mami Majdi za lektoriranje in potrpljenje.

Kazalo

Povzetek

V 18. in 19. stoletju se je razvila veja matematike, ki je postala znana kot fraktalna geometrija. Ideje in dela Benoita Mandelbrota so krepko pripomogla k njenemu hitremu razvoju in od objave njegovih del so se našli različni načini praktične uporabe tako fraktalov, kot bolj pogosto fraktalne dimenzije. Fraktalna geometrija je bila uporabljena na različnih področjih, kot so informacijska teorija, ekonomija, nevroznanost, medicina, fizika, akustika, analiza slik in drugih.

Diplomsko delo pokriva zgodovino in osnove fraktalne geometrije, opiše najbolj temeljne definicije in izreke, ki so potrebni za razumevanje področja. Očrtani so koncepti kot so mere in fraktalna dimenzija, predvsem dimenzija po metodi "štetja škatel". V nadaljevanju so opisane različne tehnike za generiranje fraktalov, med njimi je opisan tudi moj praktičen pristop k razvoju programskega orodja za generiranje fraktalov. Podala sem tudi pregled že obstoječe prosto dostopne programske opreme s to funkcionalnostjo. Naredila sem program, ki je zmožen generirati veliko različnih tipov fraktalov: IFS sistem, Mandelbrotova množica, L-sistemi, algoritem Fractal flame, fraktalna pokrajina, naključni fraktali, ...

Ključne besede : generiranje fraktalov, fraktalna grafika, fraktalno programiranje, IFS sistemi, L-sistemi, fraktalna pokrajina, naključni fraktali, prepisovalni algoritmi.

Abstract

In 18th and 19th century a new branch of mathematics developed, today known as fractal geometry. The ideas and work of Benoit Mandelbrot have been of great importance to its fast development and since the publication of his work, many practical uses of fractals and even more often fractal dimension have been found. Fractal geometry is used in many different fields such as information theory, economy, neuroscience, medicine, physics, acoustics, image analysis and other.

This thesis covers history and basics of fractal geometry, describes fundamental definitions and theorems that are necessary for understanding of the field. Concepts like measure and fractal dimension are described, especially the "box counting"method. In continuation different techniques for fractal generating are described. Along with them my practical approach to development of fractal generating software is also described and an overview of existing fractal generating software is given. I have developed computer software that is capable of generating many different types of fractals: IFS system, Mandelbrot set, L-systems, fractal flame algorithm, fractal terrain, random fractals, ...

Keywords : fractal generating, fractal graphics, fractal programming, IFS systems, L-systems, fractal terrain, random fractals, rewriting algorithms.

1 Uvod

Zgodovina fraktalne geometrije je zaznamovana z odkrivanjem lastnosti neodvedljivih, vendar povsod zveznih funkcij in krivulj, sebi podobnih množic in množic s fraktalnimi dimenzijami. Najprej so bile te raziskave in rezultati obravnavani kot anomalija in splošno mnenje je bilo, da takšne krivulje nimajo kakršnekoli praktične aplikacije v današnjem svetu. [7] Danes je to popolnoma drugače.

Prve ideje o fraktalnih množicah so se pojavile konec 19. stoletja, vendar so se le-te začele uporabljati v praksi šele v zadnjih 40 letih. Uporaba fraktalne geometrije zajema danes področje fizike, ekonomije, biologije, medicine, računalništva (kompresija slik, računalniška grafika, posebni učinki v filmih, generiranje glasbe, klasificiranje vzorcev, ...) in druga področja.

Obstaja veliko različnih načinov definiranja fraktala. Konceptualno lahko definiramo fraktal kot samo sebi podobno strukturo. [5] To pomeni, da ne glede na to, kako zelo strukturo povečamo, imamo pred sabo še vedno kopijo originalne množice. Obstajajo različni tipi samopodobnosti, vendar pri nekaterih ne dobimo vedno popolnoma enake kopije originalne množice. Zato je fraktal objekt, ki je bolj "nepravilen"kot klasični geometrijski objekti (premice, kvadrati, kocke in podobno). Tudi ko je fraktalna struktura zelo povečana, je še vedno tako kompleksna, kot prej. Torej lahko gledamo na fraktal kot na neskončno kompleksno strukturo.

Bolj kot na matematično podlago in klasifikacijo samih fraktalov, sem se po kratkem in splošnem opisu teorije fraktalov v tej diplomski osredotočila na točno določene tehnike za generiranje fraktalov. Začela sem s tehniko IFS in s primerki te tehnike (Kochova snežinka, algoritem Fractal flame in preprogo Sierpińskega). Nadaljevala sem s tehniko, ki uporablja funkcije nad kompleksnimi števili, kot so Mandelbrotova množica in z njo povezane Julia množice in Newtonovi fraktali. Opisala sem tudi L-sisteme, ki so pravzaprav paralelni rekurzivni prepisovalni sistemi in se jih uporablja za modeliranje rasti rastlin, pod to tehniko lahko razvrstimo algoritem fraktalnega drevesa. Končala sem z naključnimi fraktali, ki uporabljajo stohastična pravila in s primerki te tehnike (fraktalna pokrajina in Brownovo gibanje). [8]

Podrobno sem opisala zgoraj omenjene načine in algoritme njihove konstrukcije, ki sem jih nato vključila v namizno aplikacijo, napisano v programskem jeziku Java in jo razvila tekom pisanja te diplomske naloge. Aplikacija je inte-

raktivna in omogoča uporabniku spreminjanje določenih parametrov fraktala, nekatere povezave med določenimi tipi fraktalov so prikazane grafično. Služi lahko predvsem v izobraževalne namene.

2 O fraktalih

2.1 Kaj so

V osemdesetih letih 20. stoletja so se fraktali približali ljudem skozi zanimive barvne podobe, narisane s pomočjo računalnikov. Vendar se malokdo zaveda, na koliko fraktalnih sistemov se pravzaprav vsak dan opiramo in da so fraktali bistveno spremenili naše razumevanje sveta.

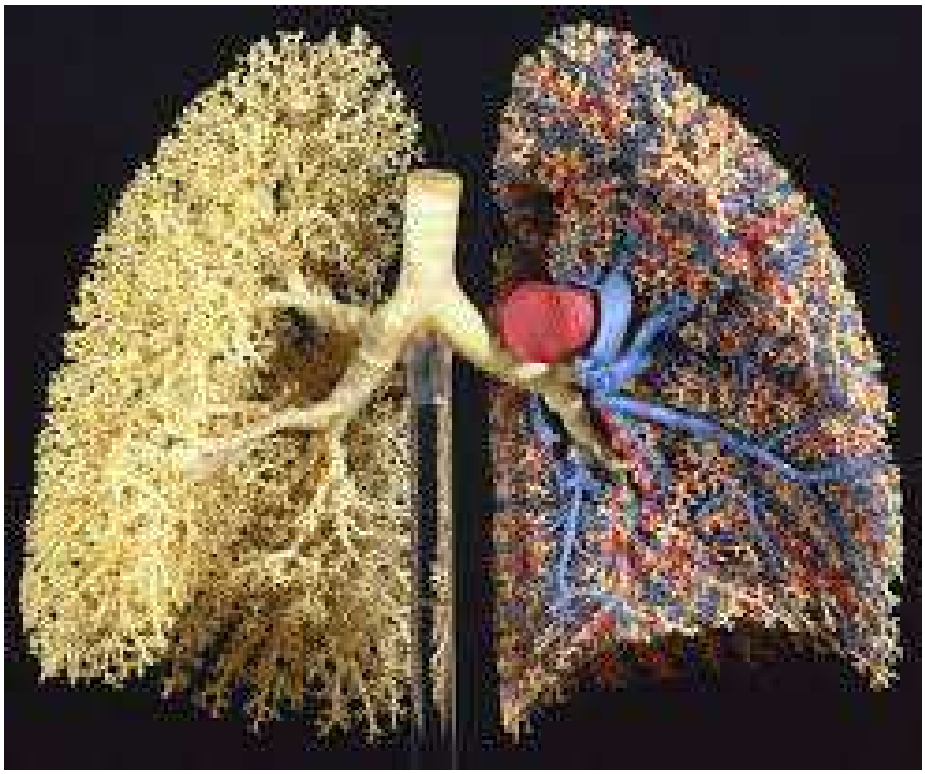
Naravni svet okoli nas je definiran z nepravilnimi površinami in oblikami z različnimi robi in grobimi koti. Vendar je klasična Evklidska geometrija opisala le idealne oblike - krog, sfera, kvadrat, kocka - te pa le redko, če sploh kdaj, najdemo v naravi. Mandelbrot v svoji knjigi Fraktalna geometrija narave (1982) piše: "Oblaki niso krogle, gore niso stožci, obale niso krogi, lubje ni gladko in strela ne potuje v ravni črti."Fraktali pa so geometrija naravnega sveta in opišejo teksturo realnosti.

Slika 1: Fraktali v naravi - cvetača

V tem smislu si jih je tudi najlažje predstavljati. Oblaki, gore, obale, cvetača,

praprot - vse to so naravni fraktali. Te oblike imajo neko skupno intuitivno aestetsko vrednost, so zapletene in nepravilne. Celo matematiki so se jih sprva izogibali, saj se je zdelo, da se jih z enačbami ne da ukrotiti oz. razložiti.

Slika 2: Fraktali v naravi - človeška pljuča



Fraktalna matematika in s tem povezana teorija kaosa navdihuje različne znanstvene discipline - medicino, genetiko, inženirstvo, umetnike in celo glasbenike. V praksi lahko najdemo fraktalno matematiko v oblikovanju realističnih računalniških grafik (pokrajine ...), kompresiji datotek, arhitekturi omrežij in celo pri diagnosticiranju nekaterih bolezni. Prav tako si lahko s fraktalno geometrijo pomagamo pri razumevanju različnih sistemov. Obsežnost in čas potresov, variacije v človeškem bitju srca, razširjenost bolezni, so le trije primeri, v katerih lahko fraktalna geometrija opiše nepredvidljivo. Tak primer je tudi finančni trg, ki ga je prav tako raziskoval Mandelbrot, ko je delal v šestdesetih

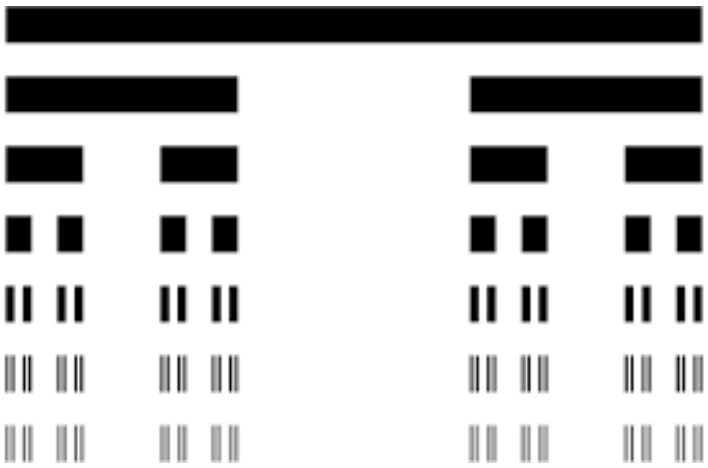
letih za IBM. Raziskoval ga je v smislu profita in izgub, ki so jih imeli trgovalci skozi čas. Ugotovil je, da se je opis s fraktalno matematiko dobro prilegal oz. opisal to dinamiko. Opozarjal je, da trgovalci veliko tvegajo, ko vzamejo v zakup, da je trg predvidljiv in imun na velika nihanja. Kljub temu, da sicer fraktalna matematika ne more predvideti velikih dogodkov v kaotičnih sistemih, nam lahko pove, da se bodo taki dogodki zgodili.

2.2 Zgodovina

Odkritje prvega matematičnega fraktala pripisujemo leta 1861 leta Karlu Weierstraussu. Našel je zvezno funkcijo, ki ni nikjer odvedljiva - krivulja, sestavljena le iz kotov, ki ji je nemogoče določiti stopnjo spremembe na katerikoli točki. Weierstraussova funkcija je bila šok matematikom njegovega časa in menili so, da nič podobnega ni moč najti v naravi.

Kmalu po Weierstraussu je Georg Cantor, izumitelj teorije množic, želel razumeti kontinuiteto in neskončnost. To ga je leta 1883 pripeljalo do fraktala, ki ga sedaj poznamo kot Cantorjeva množica. Začnemo s črto, odstranimo srednjo tretjino črte in ostaneta nam dve ravni črti. Odstranimo srednjo tretjino vsake te črte, to nadaljujemo 'do neskončnosti' in dobimo Cnatorjevo množico. V tem času Cantor tega ni imenoval 'fraktal', a je to uporabil kot primer nikjer zvezne množice, ki ni merljiva, saj bi naključno vržena puščica imela neskončno majhno možnost, da jo zadane, saj je vsak del sestavljen skoraj iz samih lukenj.

Slika 3: Cantorjeva množica



Potovanje skozi zgodovino fraktalov nas pripelje do dveh francoskih matematikov, Gastona Julia in Pierra Fatou. Okrog prve svetovne vojne sta oba, neodvisno drug od drugega, raziskovala transformacije v kompleksni ravnini. Transformacija je pravilo, ki za katerokoli točko v ravnini producira drugo točko v ravnini. To si lahko predstavljamo kot rotacije, raztegovanje, prepognjevanje ravnine, ki jo nato položimo nazaj v ravnino v novi konfiguraciji. Julia in Fatou sta bila oba zainteresirana v proces iteracije, uporabljanje rezultata ene transformacije, kot vnosnega parametra nove transformacije. To se je dogajalo v času pred modernimi računalniškimi grafikami, zato so njuni izračuni in skice nastajale ročno. Kljub temu sta našla privlačne točke v prostoru, ki so pritegnile sosednje točke k sebi, in odbijajoče točke, ki so odbijale sosednje točke. Primer tega je iterativno kvadriranje kompleksnega števila, odvisno od začetne točke, rezultat bo ali pobegnil v neskončnost ali pa se ustalil pri neki točki. Julia in Fatou sta se zavedala, da je meja med območji privlačnosti in odbojnosti zelo zapletena, vendar brez računalnikov, ki bi avtomatizirali proces računanja, nista nikoli videla lepote in daljnosežnosti svojih idej. Območje, ki je tako definirano, je danes znano kot Juliajeva množica.

Delo Julia in Fatou je bilo v veliki meri spregledano s strani matematikov, dokler ni poljski matematik Mandelbrot v poznih 70. letih 19. stoletja na novo osvetlil tega področja. V tem času je Mandelbrot delal na IBM, a je bil

tudi gostujoči profesor na Harvardu, kjer je začel svoje raziskovanje Juliajevih množic. Mandelbrot je začel z najbolj preprosto transformacijo $z = z^2 + c$, kjer je c konstanta in z koordinata točke v kompleksni ravnini. Rezultat enačbe postane nova vrednost z, ki postane vhodni parameter naslednji iteraciji, ta proces pa ponavljamo do vnaprej določenega števila iteracij. Mandelbrot je za vsako točko v kompleksni ravnini narisal črno točko za tiste točke, ki jih je privlačila druga točka in pustil prazne tiste, katerih vrednost je bila zelo velika oz. je pobegnila v neskončnost. Nadalje je Mandelbrot ustvaril še zemljevid Juliajevih množic, ki ga danes poznamo kot Mandelbrotova množica.

Koncept fraktala je bil tako na samem začetku predstavljen s strani Benoita Mandelbrota. Leta 1975 si je zamislil ime 'fraktal' kot način, kako opisati oblike, ki so enako podrobne na vseh stopnjah povečave. Kar se je začelo kot raziskovanje obskurnega področja matematike, se je s pomočjo Mandelbrota razvilo v novo področje, fraktalno geometrijo. Fraktalna geometrija je podaljšek klasične geometrije, ki lahko s pomočjo računalnikov modelira in opiše strukture od školjk do galaksij.

Termin 'fraktal', ki je bil izumljen s strani Mandelbrota v letu 1975, opiše skupek množic, ki so, kljub temu, da so jih drugi matematiki že raziskovali, do takrat ostale neimenovane. To je bilo rojstvo fraktalne geometrije. Termin je razvil iz latinske besede za 'nepravilen'. Kasneje je povedal tudi, da si fraktali zaslužijo, da jim rečemo 'geometrično kaotični'.

Dokaj vzporedno z Mandelbrotovimi odkritji je biolog Artistid Lindenmayer predstavil formalno metodo modeliranja razvoja in rasti rastlin. Danes poznani kot L-sistemi, so na nek način prepisovalni sistemi, splošno orodje za konstrukcijo kompleksnih objektov, začenši s preprostim, preko rekurzivnega zamenjevanja delov. Čeprav so osnove prepisovalnih sistemov bile poznane od začetka tega stoletja, se jih ni veliko uporabljalo do leta 1950, ko jih je Noam Chomsky uporabil za opisovanje gramatike naravnih jezikov. Ključna razlika med Chomskyevo in Lindenmayerjevo uporabo pravil je, da jih je Chomsky aplicira sekvenčno, Lindenmayer pa hkrati. Prve grafične predstavitve L-sistemov so bile leta 1975, s strani Fritjersa in Lindenmayerja. Potencial L-sistemov za produciranje realističnih slik rastlin je bil demonstriran komaj leta 1987. Leta 1979 sta Szilard in Quionton pokazala, da lahko L-sistemi generirajo fraktalne krivulje. Nadalje je bila najdena še dimenzija teh krivulj, generiranih z L-sistemi in tri dimenzionalne verzije.

Od Mandelbrotove prve knjige o fraktalih se je to področjo zelo razvilo. Danes poznamo še veliko drugih tipov fraktalov, kot so IFS sistemi. V svoji trenutni obliki so bili 'izumljeni' s strani John E. Hutchinsona v 1981 letu in popularizirani s strani Michaela Barnsleya v knjigi Fraktali vsepovsod. Poznamo pa tudi naključne fraktale.

2.3 Samopodobnost

Geometrični objekti, ki so sami sebi podobni, ko spreminjamo razdaljo na kateri jih opazujemo, so uporabni, ko želimo najti red v očitnem neredu. Tako je najbolj opazna lastnost fraktalov njihova samopodobnost, ko celota spominja na manjše dele same sebe na različnih povečavah. Ta lastnost nas spomni na naravne fraktalne strukture okoli nas, kako lahko veja z majhnimi vejicami zgleda kot večja veja, ki zgleda podobno kot celotno drevo, razbrazdana površina kamna zgleda podobno celotni gori, razvejanost večjih rek spominja na razvejanost manjših vodotokov. Naravne fraktalne oblike so zgrajene iz preprostih pravil, pogosto iz preproste enačbe, ki jo ponavljamo v neskončnost.

2.4 Fraktalna dimenzija

Naslednja ključna lastnost fraktala je njegova dimenzija. Poznana nam je že topološka dimenzija - ena sama dimenzija ravne črte, dve dimenziji kvadrata in podobnih oblik, tri dimenzije objektov in prostora okoli nas. Vendar pri bolj kompliciranih oblikah, kot so navadni Evklidski objekti, stvari niso tako preproste.

Felix Hausdorff je predstavil drugačen način pogleda na merjenje dimenzije, in sicer je predstavil koncept fraktalne dimenzije, ki ji danes rečemo tudi Hausdorffova dimenzija. Tako so nastali zanimivi koncepti, npr. 1.5 dimenzionalnih objektov - fraktali.

Da pravilno razumemo fraktalno dimenzijo, moramo najprej razumeti splošen koncept merjenja dimenzij pri bolj zapletenih objektih. Kocka, ki ima tri dimenzije, je lahko razrezana v 8 kock, ki so polovične velikosti originalne kocke. To lahko posplošimo kot razdeljevanje n-dimenzionalne oblike na m^n kopije same sebe, ki so po velikosti $1/m$, velikosti originalne oblike.

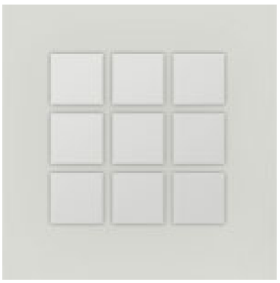
Če lahko preštejemo število kopij samega sebe, ki jih objekt vsebuje in vemo kakšne velikost so, lahko iz tega izvemo njegovo dimenzijo s pomočjo enačbe:

$$dimenzija = \frac{log(m^n)}{-log(1/m)}$$

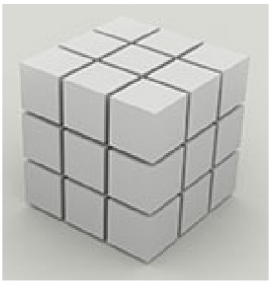
Slika 4: Faktalna dimenzija

Primer: Vsako stran delimo s 3

Slika 5: 3 kopije
$m^n = 3^1 = 1$



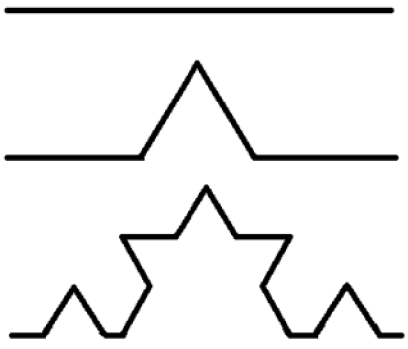
Slika 6: 9 kopij
$m^n = 3^2 = 9$



Slika 7: 27 kopij
$m^n = 3^3 = 27$

Klasičen primer sami sebi podobne krivulje s fraktalno dimenzijo je Kochova krivulja. Prvič je bila predstavljena s strani Helge von Koch v letu 1904, krivulja je del družine krivulj, ki ji matematiki pravijo kar 'patologična', je neskončno dolga, a ima končno površino.

Slika 8: Koraki razvijanja Kochove krivulje



Pri vsakem koraku je vsaka ravna črta zamenjana z osnovno krivuljo. Tako dobimo štiri kopije velikosti 1/3 prejšnje krivulje. Če to vstavimo v našo formulo za fraktalno dimenzijo:

$$\frac{log(4)}{-log(1/3)} = 1.26$$

Slika 9: Faktalna dimenzija Kochove krivulje

3 Raziskovalna metodologija in pristop k problemu

V svojem delu sem se osredotočila na računalniško grafiko in vzorce, ki jih lahko z digitalno tehniko in s pomočjo algoritmov narišemo na ekran. Obstaja že kar nekaj računalniških programov, ki nam pomagajo pri vizualizaciji različnih fraktalov in s tem fraktalnih tehnik. Moja namizna aplikacija je nastajala za izobraževalne namene, skozi njeni razvoj in nastanek sem želela podrobneje spoznati to zanimivo vejo geometrije, ki je tekom študija nismo veliko omenjali. Skozi manipulacijo parametrov posameznih fraktalov, ki jo bo moja aplikacija omogočala, se bo mogoče tudi poučiti in spoznavati, kako se strukture odzivajo na spremembe parametrov.

Za razvoj aplikacije sem uporabila razvojno orodje Netbeans in programski jezik Java.

4 Pregled že obstoječe programske opreme za vizualizacijo fraktalov

Za generiranje fraktalov obstaja veliko plačljivih in neplačljivih programov, izbrala sem le nekaj bolj uporabljenih in tiste, ki so me prevzeli in navdihnili.

4.1 FractInt

Je najstarejši aktivni in hkrati prevladujoči program na tem področju. Lahko ga dobimo tako za operacijski sistem Windows (predvsem starejše verzije) kot tudi Linux. Sposoben je generirati Mandelbrotovo množico, Juliajevo množico, Newtonove fraktale, Henonove, Lorenzove in Rosslerjeve atraktorje, Lyapunov fraktal, IFS sisteme, L-sisteme in celične avtomate. Če želimo pisati lastne formule, zna brati tudi skripte. Za uporabo na operacijskem sistemu Windows si moramo omisliti emulator ali virtualno okolje, saj je bil program napisan za sistem DOS in ga v novejših verzijah operacijskega sistema Windows ne moremo zagnati.

Domača stran: `http://www.fractint.org/`

4.2 Ultra Fractal

Najbolj popularen izmed plačljivih programov s preizkusno verzijo, ki traja 30 dni. Dobimo ga lahko za operacijski sistem Windows in Mac OS X. Sposoben je generirati veliko predpripravljenih fraktalov, od različnih variacij Mandelbrotovih množic, Juliajevih množic, Newtonovih fraktalov do trikotnika Sierpińskega. Z njim lahko pišemo tudi skripte v posebnem jeziku. S temi skriptami lahko generiramo praktično katerokoli vrsto fraktala, ga pobarvamo z lastnim algoritmom za barvanje in ga transformiramo z lastno transformacijo. Z njim lahko kreiramo tudi animacije.

Domača stran: `http://www.ultrafractal.com/`

4.3 ChaosPro

ChaosPro ponuja veliko različnih možnosti. Z njim lahko generiramo IFS fraktale, L-sisteme, z njim lahko pišemo lastne skripte, brskamo lahko po zajetni

knjižnici fraktalnih enačb, ustvarjamo animacije, sestavljamo 3D fraktale in drugo. Program lahko uporablja skripte, napisane tako za FractInt kot tudi Ultra Fractal, zato se mi zdi zelo dobra izbira.

Domača stran: `http://www.chaospro.de/`

4.4 Apophysis

Apophysis je urejevalnik fraktalnih plamenov, ki so "podaljški"IFS sistemov. Z njim lahko generiramo IFS fraktale, urejamo že obstoječe iz zajetne knjižnice, nanje apliciramo variacije, eno ali več hkrati, lahko pa variacije tudi animiramo. Je brezplačen in odprtokoden.

Domača stran: `http://www.apophysis.org/`

4.5 GIMP

Za odprtokoden program za urejanje slik GIMP obstaja več dodatkov. Z njihovo pomočjo zna tudi GIMP generirati fraktale. Med drugimi sta taka dodatka Flame in IFS Fractal. Flame zmore generirati naključne IFS fraktale, preko katerih lahko spustimo razne variacije. IFS Fractal pa lahko generira samo IFS fraktale, vendar lahko nadziramo formule, s katerimi nastanejo, kar s Flamu ne moremo. GIMP obstaja za operacijski sistem Windows, Linux in Mac OS X.

Domača stran: `http://www.gimp.org/`

4.6 FRAX

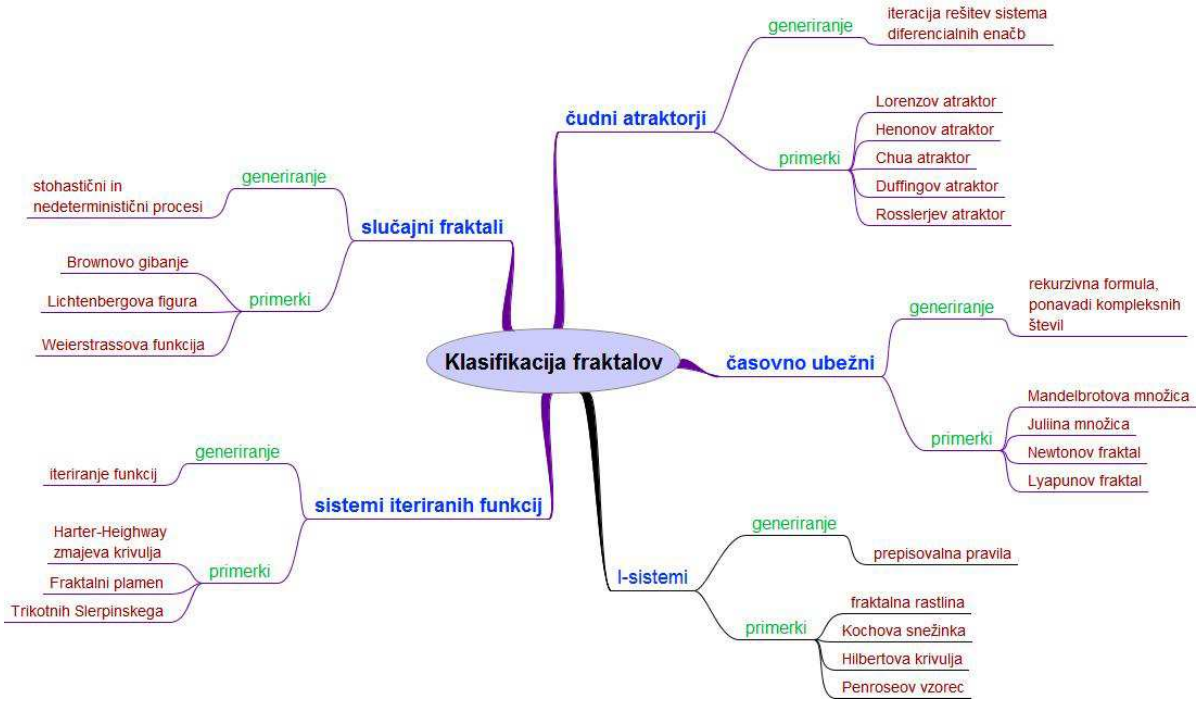
Zadnji, rahlo netipični član opisane družine programske opreme, je iOS aplikacija FRAX, ki jo lahko dobimo zelo poceni. Je bolj umetniško usmerjena, saj nadziramo fraktalne oblike, barve in teksture s prsti po ekranu telefona in ne zahteva toliko matematičnega znanja.

Domača stran: `http://fract.al/`

5 Tehnike za generiranje fraktalov in pripadajoči primeri

Na sliki 10 sem v obliki miselnega vzorca orisala klasifikacijo fraktalov in primerke vsake skupine, med katerimi jih bo kar nekaj vidnih tudi v moji izvedbi aplikacije. Kot lahko vidimo, obstaja 5 glavnih tehnik konstrukcije fraktalov, katerim pripadajo posamezni primerki, ki so si s svojo različnostjo pridobili svoje ime.

Slika 10: Klasifikacija fraktalov



5.1 IFS sistemi

IFS je kratica za 'Iterated Function System' ali sistemi iteriranih funkcij. Fraktale tega tipa dobimo z uporabo različnih funkcij. Ti fraktali so unija kopij samega sebe, vsaka kopija je transformirana z neko afino transformacijo oz. funkcijo. Funkcije običajno krčijo, tako da se točke približujejo druga drugi

(delajo vedno manjše like), vse do neskončno majhnih oblik. Iz tega tudi izvira samopodobnost teh fraktalov.

Vsaka funkcija, ki jo uporabimo, je nov atraktor na končni sliki. Oblika funkcij je navadno dana v obliki koeficientov od a do f, ki edinstveno odločajo afino transformacijo. Da dobimo določeno obliko, lahko uporabimo več atraktorjev (več afinih transformacij).

$$\mathrm{F}_i(x, y) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

Slika 11: Uporaba afinih transformacij

Najbolj pogostemu algoritmu za računanje IFS fraktalov rečemo "igra kaosa". Izberemo si naključno točko v naši ravnini in iterativno apliciramo naključno izbrano funkcijo iz našega nabora funkcij. Izhodi iz funkcij so točke, ki so v množici točk, ki definirajo naš fraktal in te tudi narišemo na ekran. Tak algoritem generira popolnoma vse točke fraktala. Izkaže se tudi, da izbira začetne točke ne vpliva toliko na obliko fraktala.

Algoritem 1 Risanje IFS fraktala

```
1: procedure IgraKaosa(steviloIteracij, steviloFunkcij)
2:     (x, y) = naključna točka v kvadratu
3:     while k ≤ steviloIteracij do
4:         i = naključno število od 0 do steviloFunkcij
5:         (x, y) = F_i(x, y)
6:         plot(x, y) razen ko k ≤ 20
7:         k = k + 1
```

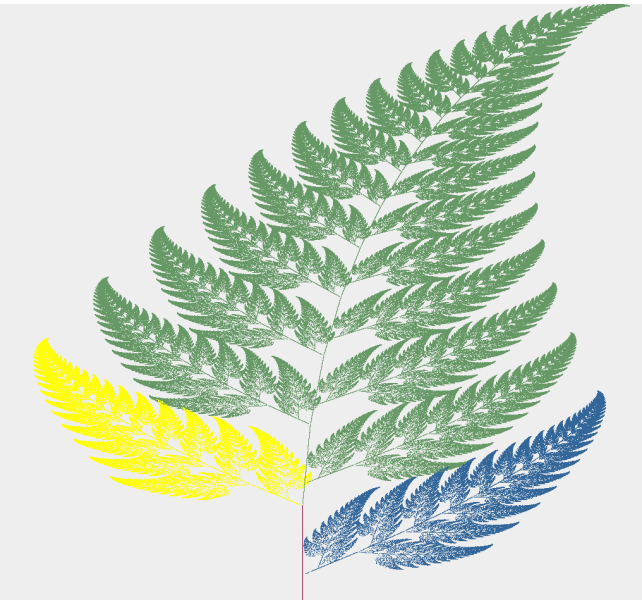
5.1.1 Barnsleyevo pero

Za Barnsleyevo pero, ki ga lahko vidimo na sliki 12, sem uporabila štiri funkcije z različnimi verjetnostmi. Matrika koeficientov teh funkcij, s katerimi sem ga generirala, je naslednja:

Tabela 1: Matrika za konstrukcijo Barnsleyevega peresa

funkcija	a	b	c	d	e	f	verjetnost	barva
f1	0	0	0	0.16	0	0	0.01	rdeča
f2	0.85	0.04	-0.04	0.85	0	1.6	0.85	zelena
f3	0.2	-0.26	0.23	0.22	0	1.6	0.07	rumena
f4	-0.15	0.28	0.26	0.24	0	0.44	0.07	modra

Slika 12: Barnsleyevo pero



Dele peresa sem pobarvala glede na to, katera funkcija je definirala izbrano točko na ekranu. Tako lahko vidimo, da funkcija, ki je bila aplicirana z največjo verjetnostjo, generira večino izrastkov iz peclja, ostali dve generirata dva korenska izrastka, tista z najmanjšo pa le pecelj peresa.

5.1.2 Trikotnik Sierpińskega

Trikotnik Sierpińskega ima obliko trikotnika, kot nam pove že ime. Prav tako je rekurzivno sestavljen iz vedno manjših trikotnikov. V osnovi je bil skonstruiran

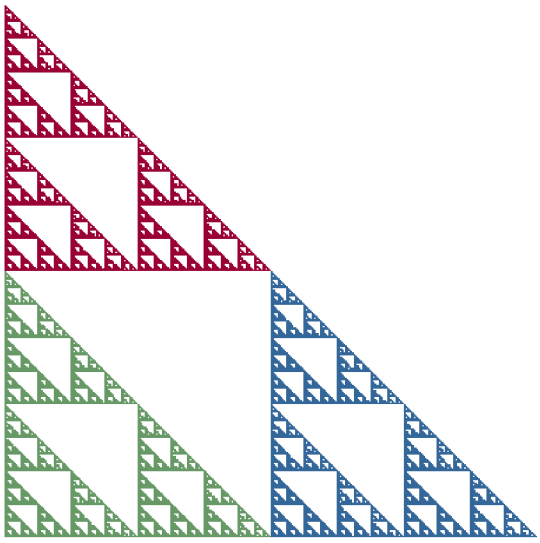
kot krivulja in je eden izmed najbolj osnovnih primerov sami sebi podobnih množic, ki jih lahko matematično reproduciramo na katerikoli povečavi. Imenovan je po poljskem matematiku Wacławu Sierpińskemu, vendar se je pojavljal že stoletja pred delom Sierpińskega.

Obstaja več načinov konstrukcije, lahko bi ga skonstruirali kot L-sistem, sama pa sem si izbrala IFS način konstrukcije. Matrika koeficientov funkcij je tako naslednja:

Tabela 2: Matrika za konstrukcijo trikotnika Sierpińskega

funkcija	a	b	c	d	e	f	barva
f1	0.5	0	0	0.5	0	0	zelena
f2	0.5	0	0	0.5	0.5	0	modra
f3	0.5	0	0	0.5	0	0.5	rdeča

Slika 13: Trikotnik Sierpińskega



5.1.3 Fraktalni plameni

Fraktalni plameni so zelo splošen pojem. Nastanejo ne le na podlagi afinih transformacij, temveč tudi nelinarnih. To naredimo tako, da ob vsaki iteraciji naključno izbrane ene izmed afinih transformacij, apliciramo še vnaprej izbrano nelinearno funkcijo, ki ji rečemo variacija. Lahko apliciramo tudi več nelinearnih funkcij, povrhu še z različnimi verjetnostmi, vendar sem v svojih primerih zaradi nazornosti uporabila le eno hkrati.

$$F_i(x, y) = V(a_i x + b_i y + c_i, d_i x + e_i y + f_i)$$

Slika 14: Uporaba variacije na afinih transformacijah

Pri fraktalnih plamenih je potrebno tako kodo za generiranje IFS fraktalov dopolniti na naslednji način:

Algoritem 2 Risanje fraktalnih plamenov

```
1:  procedure IgraKaosa(steviloIteracij, steviloFunkcij, V)
2:      (x, y) = naključna točka v kvadratu
3:      while k ≤ steviloIteracij do
4:          i = naključno število od 0 do steviloFunkcij
5:          (x, y) = F_i(x, y)
6:          (x, y) = V(x, y)
7:          plot(x, y) razen ko k ≤ 20
8:          if k ≠ 20 then
9:              plot(x, y)
10:         k = k + 1
```

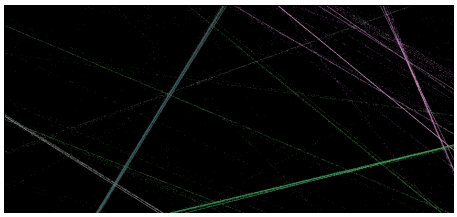
Za naslednje primere na slikah sem uporabila 4 funkcije (afine transformacije), definirane z naslednjo matriko koeficientov:

Tabela 3: Matrika afinih transformacij fraktalnih plamenov na slikah

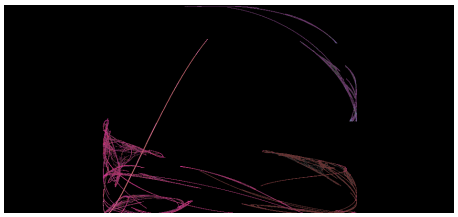
funkcija	a	b	c	d	e	f
f1	0.2	1	0.7	0	0.9	0.792
f2	0.954	1	1.095	0.719	0.208	1.089
f3	-0.5	1.439	0.903	1	0.393	1.473
f4	0.5	0.495	1.171	-1	0.98	1.229

Nastale slike sem ločila po uporabljeni variaciji. Za vsako sliko podam formulo, ime ter definiram spremenljivke. Pod slikami so podane enačbe $u(x, y)$ in $w(x, y)$, ki jih interpretiramo v kontekstu naslednje definicije variacije:

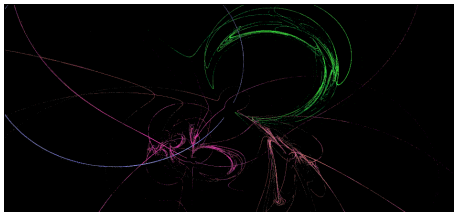
$$V(x, y) = (u(x, y), w(x, y))$$



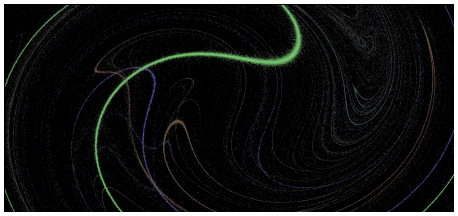
Slika 15: Linearna variacija
$u(x, y) = x$
$w(x, y) = y$



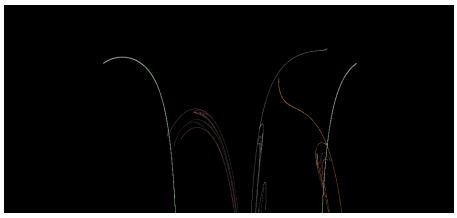
Slika 16: Sinusna variacija
$u(x, y) = sin(x)$
$w(x, y) = sin(y)$



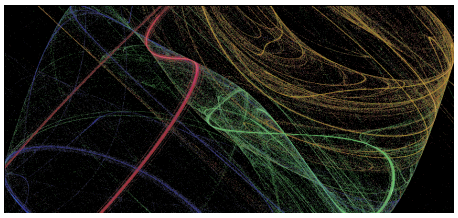
Slika 17: Sferična variacija
$r = \frac{1.0}{x^2+y^2}$
$u(x, y) = rx$
$w(x, y) = ry$



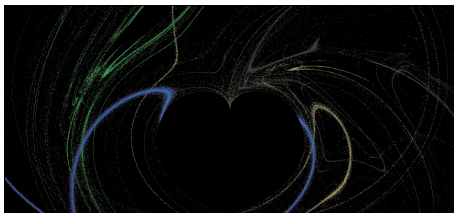
Slika 18: Vrtinčasta variacija
$r = x^2 + y^2$
$u(x, y) = x * sin(r) - y * cos(r)$
$w(x, y) = x * cos(r) + y * sin(r)$



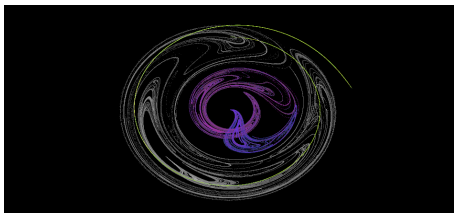
Slika 19: Polarna variacija
$u(x, y) = \frac{atan(y,x)}{\pi}$
$w(x, y) = \sqrt{(x^2 + y^2)} - 1.0$



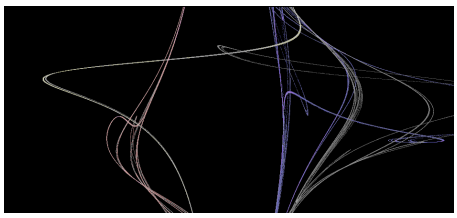
Slika 20: Variacija robčka
$r = \sqrt{(x^2 + y^2)}$
$theta = atan(y, x)$
$u(x, y) = r * sin(theta + r)$
$w(x, y) = r * cos(theta - r)$



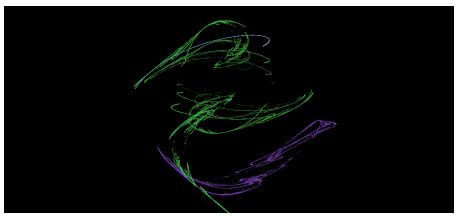
Slika 21: Variacija srca
$r = \sqrt{(x^2 + y^2)}$
$theta = atan(y, x)$
$u(x, y) = r * sin(theta * r)$
$w(x, y) = -r * cos(theta * r)$



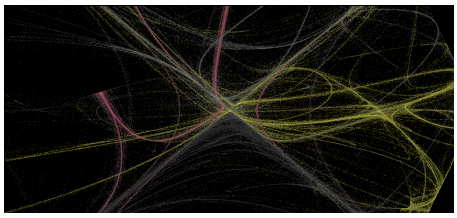
Slika 22: Variacija diska
$r = \pi\sqrt{(x^2 + y^2)}$
$theta = \frac{atan(y,x)}{\pi}$
$u(x, y) = theta * sin(r)$
$w(x, y) = theta * cos(r)$



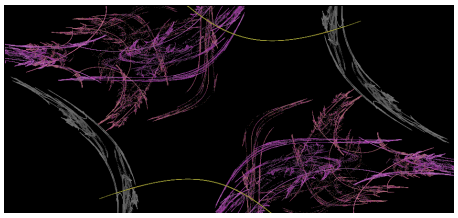
Slika 23: Variacija hiperbole
$r = \sqrt{x^2 + y^2}$
$theta = atan(y, x)$
$u(x, y) = \frac{sin(theta)}{r}$
$w(x, y) = r * cos(theta)$



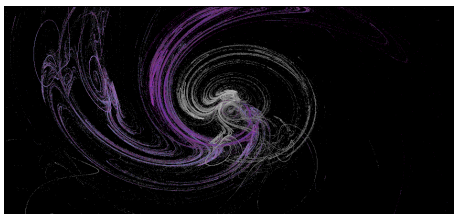
Slika 24: Variacija diamanta
$r = \sqrt{x^2 + y^2}$
$theta = atan(y, x)$
$u(x, y) = sin(theta) * cos(r)$
$w(x, y) = cos(theta) * sin(r)$



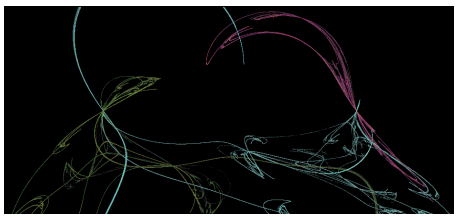
Slika 25: Variacija x
$r = \sqrt{x^2 + y^2}$
$theta = atan(y, x)$
$p_0 = sin(theta + r)^3$
$p_1 = cos(theta - r)^3$
$u(x, y) = r * (p_0 + p_1)$
$w(x, y) = r * (p_0 - p_1)$



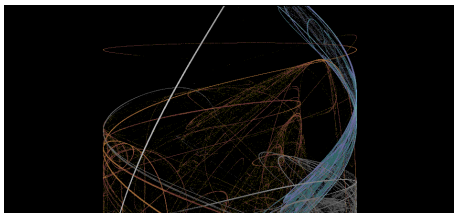
Slika 26: Variacija Julia
$r = \sqrt{x^2 + y^2}$
$theta = 0.5 * atan(y, x)$
$u(x, y) = r * cos(theta)$
$w(x, y) = r * sin(theta))$



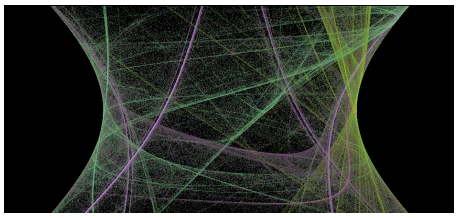
Slika 27: Eksponentna variacija
$u(x, y) = e^{(x-1.0)} cos(\pi * y)$
$w(x, y) = e^{(x-1.0)} cos(\pi * y)$



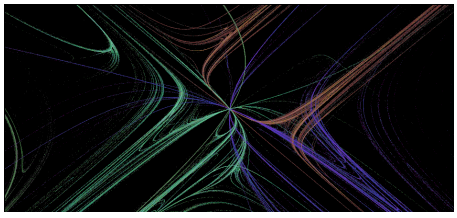
Slika 28: Variacija potenc
$r = \sqrt{x^2 + y^2}$
$theta = atan(y, x)$
$u(x, y) = cos(theta) * r^{sin(theta)}$
$w(x, y) = sin(theta) * r^{sin(theta)})$



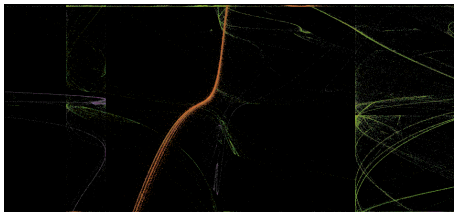
Slika 29: Variacija cilindra
$u(x, y) = sin(x)$
$w(x, y) = y$



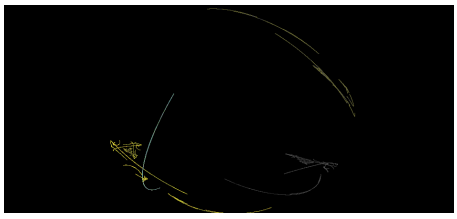
Slika 30: Variacija tangente
$u(x, y) = \frac{sin(x)}{cos(y)}$
$w(x, y) = tan(y)$



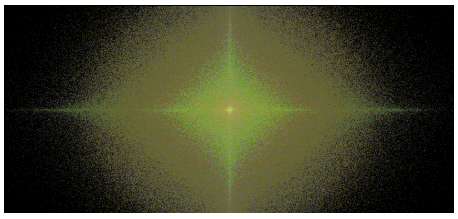
Slika 31: Variacija križa
$r = \sqrt{\frac{1.0}{(xx-yy)(xx-yy)}}$
$u(x, y) = x * r$
$w(x, y) = y * r$



Slika 32: Variacija collatz
$u(x, y) = 0.25*(1.0+4.0x-(1.0+2.0x)*$
$cos(\pi * x))$
$w(x, y) = 0.25 * (1.0 + 4.0y - (1.0 +$
$2.0y) * cos(\pi * y))$



Slika 33: Variacija mehurčka
$r = 4 + x^2 + y^2$
$u(x, y) = \frac{4.0*x}{r}$
$w(x, y) = \frac{4.0*y}{r}$



Slika 34: Variacija šuma
$theta = rand(0, 1)$
$r = rand(0, 1)$
$u(x, y) = theta * cos(2 * \pi * r)$
$w(x, y) = theta * sin(2 * \pi * r)$

5.2 L-sistemi

Predstavljeni in razviti so bili leta 1968 s strani Aristida Lindenmayerja, po katerem so dobili tudi ime. Lindenmayer je bil madžarski teoretični biolog in botanik na univerzi v Utrechtu. Z L-sistemi je opisoval obnašanje celic rastlin in modeliral morfologijo različnih organizmov. Lahko jih uporabimo tudi za generiranje fraktalov. Pri tem želim poudariti, da se da marsikaterega od teh fraktalov generirati tudi kot IFS sistem.

L-sistemi ali Lindenmayerjevi sistemi sestojijo iz kopice prepisovalnih pravil v obliki formalne gramatike. Natančneje je L-sistem sestavljen iz abecede simbolov, pri čemer vsakemu pripada zbirka pravil, po katerih se ta simbol prepiše v daljšo sekvenco simbolov. Produkcijo zmeraj začnemo z začetno besedo ali aksiomom, iz katerega nato s prepisovalnimi pravili dobimo končno besedo. Gremo lahko do poljubne globine rekurzije oz. števila iteracij, odvisno od izbrane implementacije. Izbrala sem si iterativni pristop, saj je hitrejši.

Najpogosteje uporabljamo za L-sisteme naslednjo abecedo: 'FG+-[]', ki pomeni:
F: Nariši črto in se pomakni naprej
G: Pomakni se naprej (brez risanja črte)
+: Obrni se desno za vrednost kota
-: Obrni se levo za vrednost kota
[: Shrani trenutno lokacijo
]: Obnovi prejšnjo lokacijo

Sekvenco simbolov rišemo tako, da si izberemo začetno točko na mreži, ki definira naš risalni prostor, ko pa naletimo na simbol, ki rotira smer risanja, spremenimo smer in rišemo znova naprej. Medtem se struktura lahko še razveja s simboli, ki nam velevajo, naj shranimo lokacijo na katero se bomo kasneje vrnili. Postopek ponavljamo vse dokler nismo narisali vseh simbolov sekvence, ki smo jo predhodno razvili iz aksioma s prepisovalnimi pravili.

Algoritem 3 Razvijanje besede, ki definira L-sistem

```
1: procedure GENERIRAJ BESEDO(steviloIteracij, aksiom)
2:     beseda = aksiom
3:     for k = 0 do steviloIteracij do
4:         novaBeseda =''
5:         for vsak znak c v beseda do
6:             if c ni konstanta then
7:                 novaBeseda += uporabi_prepisno_pravilo(c)
8:             else
9:                 novaBeseda+ = c
10:        beseda = novaBeseda
```

Algoritem 4 Risanje L-sistema

```
1: procedure NARIŠI BESEDO(kot, povečava)
2:     xNovi = 0, yNovi = 0, x = 0, y = 0
3:     for vsak znak c v beseda do
4:         if c ni konstanta then
5:             xNovi = x + povečava * sin(kot)
6:             yNovi = y + povečava * cos(kot)
7:             nariši_točko(xNovi, yNovi)
8:         else if c je '+' then
9:             kot+ = kot
10:        else if c je '−' then
11:            kot− = kot
12:        x = xNovi, y = yNovi
```

5.2.1 Zmajeva krivulja

Vsaka iteracija konstrukcije zmajeve krivulje vsebuje natanko dve kopiji prejšnje iteracije. Od Kochove snežinke ali trikotnika Sierpińskega se razlikuje prav po tem - njena samopodobnost je drugačna, saj na različnih stopnjah razvitosti fraktala ne izgleda popolnoma enako, vendar drži ves čas enako obliko. Skonstruiramo jo lahko kot L-sistem ali IFS sistem.

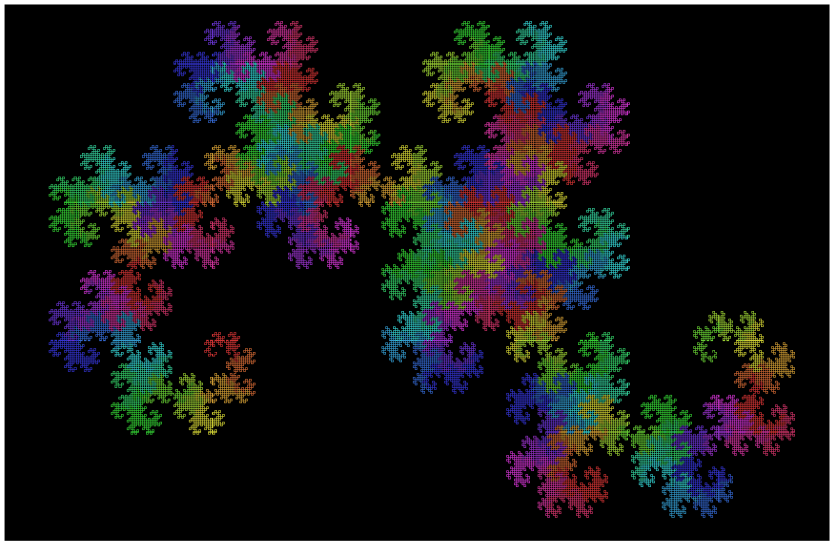
Zmajevo krivuljo sem barvala ciklično skozi število iteracij in nastala je zelo zanimiva barvna struktura. Z drugimi besedami povedano, sekvenca barv se je

na določeno število iteracij risanja besede, ki definira naš L-sistem, ponovila.

Tabela 4: Prepisovalna pravila Zmajeve krivulje

Spremenljivke:	F, X
Aksiom:	FX
Pravila:	(X = +FXFY+), (Y = FX++FY)
Kot:	45 stopinj

Slika 35: Zmajeva krivulja po 16. iteracijah



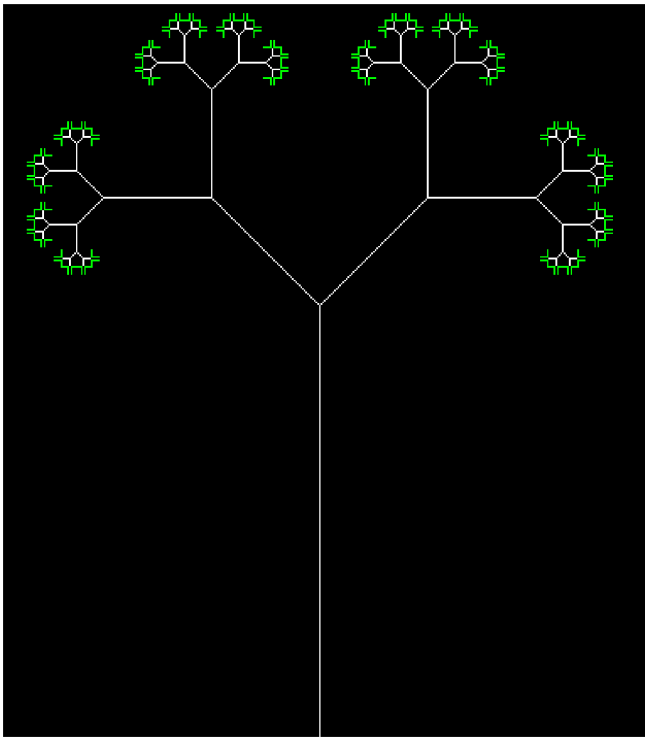
5.2.2 Pitagorovo drevo

Obstaja več različnih načinov konstrukcije Pitagorovega drevesa, ki ga lahko vidimo na sliki 36. Izbrala sem si konstrukcijo s prepisovalnimi pravili, opisanimi v naslednji tabeli:

Tabela 5: Prepisovalna pravila Kochove krivulje

Spremenljivke:	F, X
Konstante:	[ ]
Aksiom:	F
Pravila:	(X = XX), (F = X[F]F)
Kot:	45 stopinj

Slika 36: Pitagorovo drevo



5.2.3 Kochova snežinka

Švedski matematik Helge von Koch je davnega leta 1904 prvi opisal Kochovo krivuljo, matematično krivuljo, po njem je dobila tudi ime.

Kochova snežinka nastane iz Kochove krivulje. Začnemo z ravno črto, ki jo razdelimo v tri enake dele. Sredinski del zamenjamo z dvema črtama, enako

dolgima kot črta, ki smo jo zamenjali. Postavimo ju pod kotom 60, druga proti drugi tako, da se dotikata vsaka po en segment začetne ravne črte. Postopek ponavljamo na vsaki nastali črti. Po večih ponovitvah zgleda snežinka kot končen fraktal zaradi končne resolucije našega ekrana. Krivuljo se da skonstruirati tudi kot IFS sistem.

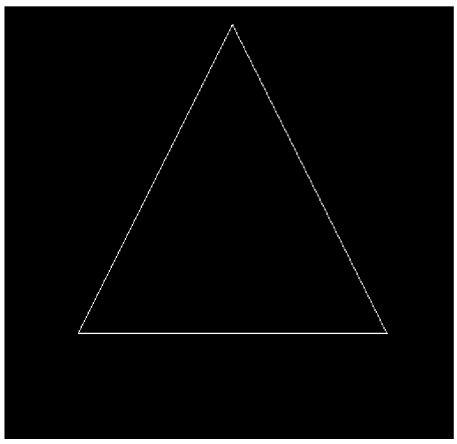
Kochova krivulja je neskončne dolžine. Z vsakim korakom konstrukcije se njena dolžina podaljša za približno 3/4, kar ne le, da ni težko dokazati, je tudi lahko dojemljivo.

Kochovo krivuljo lahko opišemo z naslednjimi pravili:

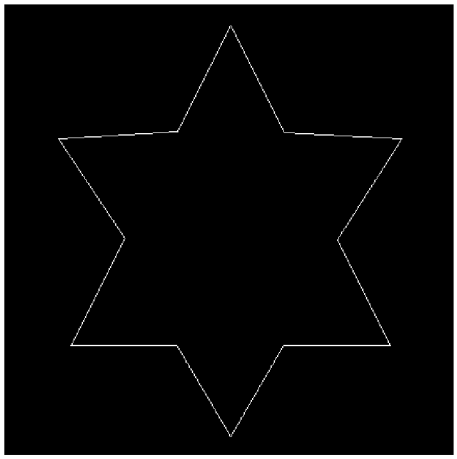
Tabela 6: Prepisovalna pravila Kochove krivulje

Spremenljivke:	F
Konstante:	+ -
Aksiom:	F++F++F
Pravila:	F = F-F++F-F
Kot:	60 stopinj

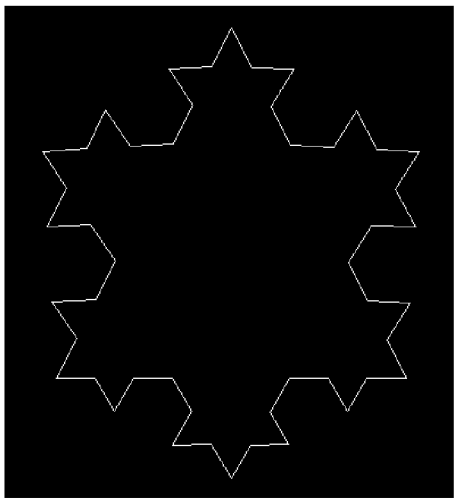
V aplikaciji sem dodala možnost spreminjanja stopnje razvitosti fraktala, kar je tudi nakazano s spodnjimi slikami.



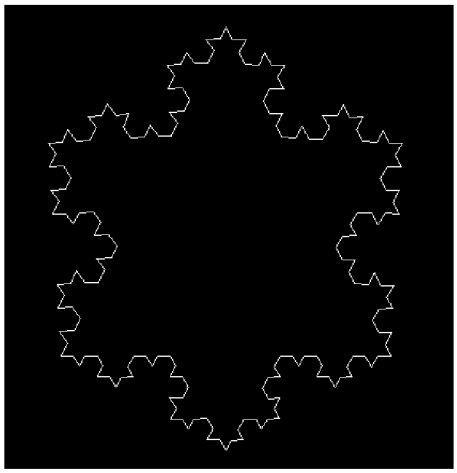
Slika 37: Kochova snežinka po 1. iteraciji



Slika 38: Kochova snežinka po 2. iteraciji



Slika 39: Kochova snežinka po 3. iteraciji



Slika 40: Kochova snežinka po 4. iteraciji

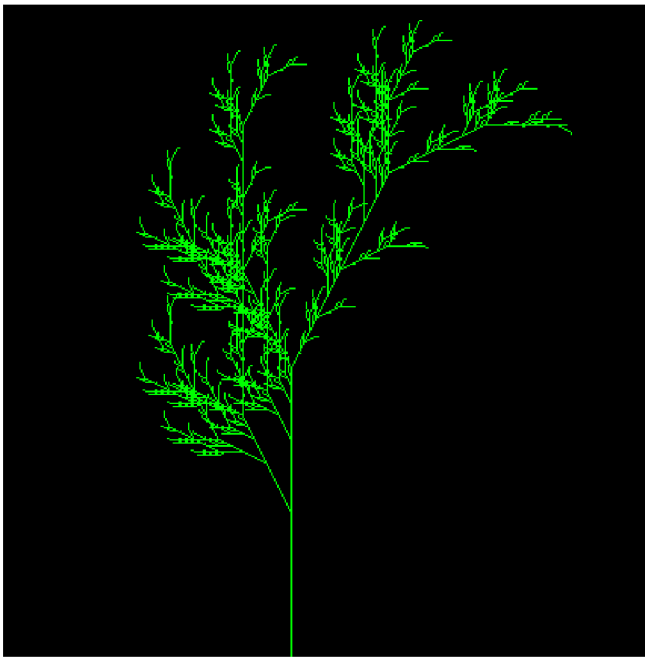
5.2.4 Fraktalna rastlina

Preprosta fraktalna rastlina, ki jo lahko vidimo na sliki 41, je nastala s pomočjo naslednjih prepisovalnih pravil:

Tabela 7: Prepisovalna pravila za fraktalno rastlino

Spremenljivke:	X, F
Konstante:	+ - [ ]
Aksiom:	X
Pravila:	(X = F-[[X]+X]+F[+FX]X), (F = FF)
Kot:	25 stopinj

Slika 41: Fraktalna rastlina



5.3 Časovno-ubežni fraktali

5.3.1 Mandelbrotova množica

Mandelbrotova množica je poimenovana po znanstveniku Benoitu Mandelbrotu, ki jo je prvi opisal. Je najbolj znan fraktal, bila je prva, ki so ji dali takšno ime.

Za nastanek Mandelbrotove množice mapiramo x in y koordinate naše risalne površine v množico kompleksnih števil c, pri čemer koordinate pikslov postanejo x in y koordinate te množice. Ta kompleksna števila vedno znova pošljemo v funkcijo predstavljeno pod sliko 42, pri čemer so začetna števila $z_0 = 0$ in barvamo piksle, pri katerih rezultat ne limitira k neskončnosti. Barvamo jih glede na to, kako hitro absolutne vrednosti rezultata funkcije pobegnejo preko števila 2. Če se to zgodi, pomeni, da Mandelbrotovi množici ne pripadajo.

Algoritem 5 Risanje Mandelbrotove množice

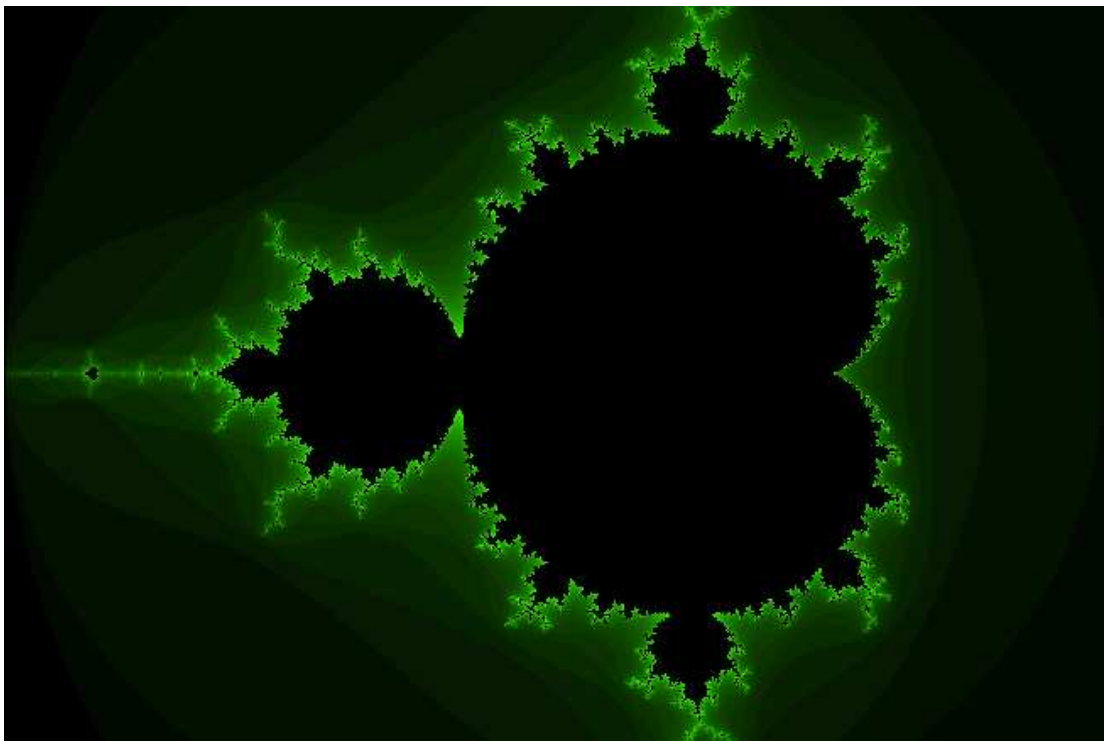
```
1: procedure RIŠI(steviloIteracij)
2:     (re_0, im_0) = (0, 0)
3:     for vsak (x, y) do
4:         (re_1, im_1) = mapiraj_parameter_v_kompleksno_ravnino(x, y)
5:         for k = 0 do steviloIteracij do
6:             (re_0, im_0) = (re_0, im_0)^2 + (re_1, im_1)
7:             if abs(re_0, im_0) > 2 then break
8:         barva = pridobi_barvo(k)
9:         pobarvaj_piksel(x, y, barva)
```

Mandelbrotova množica je povezana z Juliajevo množico, opisano v naslednjem poglavju, ki pravi, da za vse c, ki pripadajo Mandelbrotovi množici, obstaja povezana Juliajeva množica (je iz enega samega dela).

$$z_0 = 0; z = (z^2 + c)$$

Slika 42: Funkcija za generiranje Mandelbrotove množice

Slika 43: Mandelbrotova množica



5.3.2 Juliajeva množica

Juliajeve množice so generirane tako, da mapiramo koordinatni sistem podlage na katero rišemo, na množico kompleksnih števil. Pri tem koordinate pikslov, mapirane na interval od približno -2 do 2, postanejo x in y koordinate teh kompleksnih števil. Nastala množica je množica začetnih kompleksnih števil z_0.

To kompleksno število z, ki predstavlja posamezni piksel, nato vedno znova pošljemo v funkcijo, predstavljeno pod sliko 44. C predstavlja kompleksno število, ki nam določa obliko dobljene Juliajeve množice. Prav zaradi spreminjanja tega parametra dobimo veliko različnih možnih slik. Postopek ponavljamo tako dolgo, dokler z ne konvergira k številu dva, ali pa pobegne v neskončnost. Glede na število h kateremu konvergira, pobarvamo piksle.

Ta postopek apliciramo na vsak piksel posebej, tako nastane naša končna frak-

$$z = z^2 + c$$

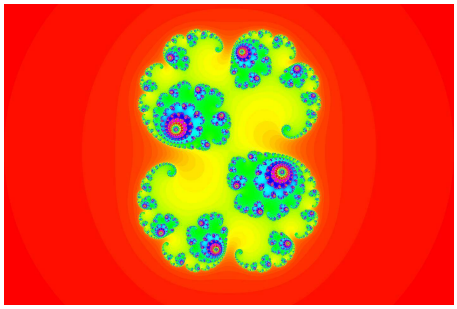
Slika 44: Funkcija za generiranje Juliajevih množic

talna slika.

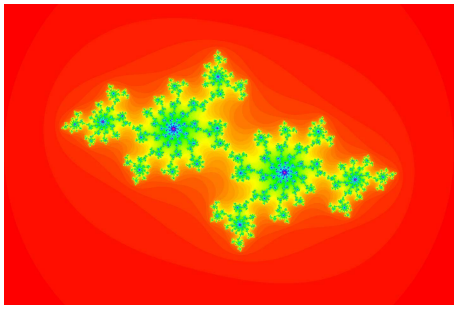
Algoritem 6 Risanje Juliajeve množice

```
1: procedure RIŠI(parameterC, steviloIteracij)
2:     for vsak (x, y) do
3:         (re, im) = mapiraj_koordinati_v_kompleksno_ravnino(x, y)
4:         for k = 0 do steviloIteracij do
5:             (re, im) = (re, im)^2 + parameterC
6:             if abs(re, im) > 2 then break
7:         barva = pridobi_barvo(k)
8:         pobarvaj_piksel(x, y, barva)
```

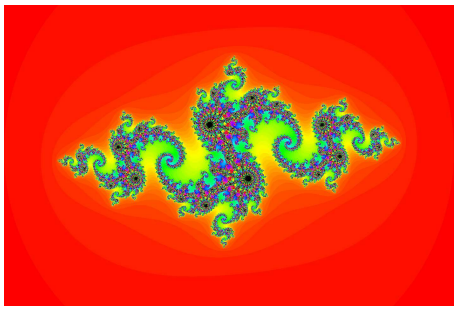
V aplikaciji imamo možnost spreminjanja kompleksnega parametra c, tako da lahko opazujemo, kako se glede na njega slika spreminja.



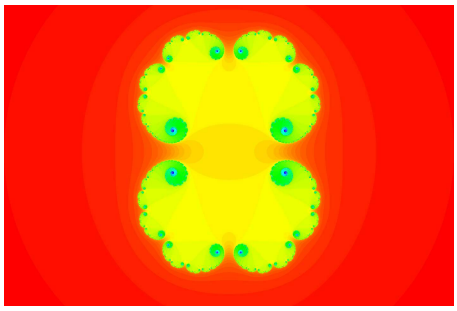
Slika 45: Juliajev fraktal za parameter $0.285 + 0.01i$



Slika 46: Juliajev fraktal za parameter $-0.70176 - 0.3842i$



Slika 47: Juliajev fraktal za parameter $-0.8 + 0.156i$



Slika 48: Juliajev fraktal za parameter $0.285 + 0i$

5.3.3 Newtonovi fraktali

Newtonovi fraktali nastanejo, ko apliciramo Newtonovo metodo za iskanje ničel polinoma na vrednosti v kompleksni ravnini. Te vrednosti nastavimo kot začetno vrednost Newtonove metode, le-ta pa nato konvergira k eni izmed ničel, če se ne pojavijo cikli. Kompleksna ravnina se tako razdeli na regije, pri čemer je vsaka asociirana z ničlo polinoma. Newtonov fraktal nam tako lahko grafično lepo prikaže, kako zelo je Newtonova metoda občutljiva na izbrano začetno točko, če se odločimo fraktal pobarvati glede na ničlo, h kateri točke konvergirajo.

$$y_{x+1} = x + \frac{f(x)}{f'(x)}$$

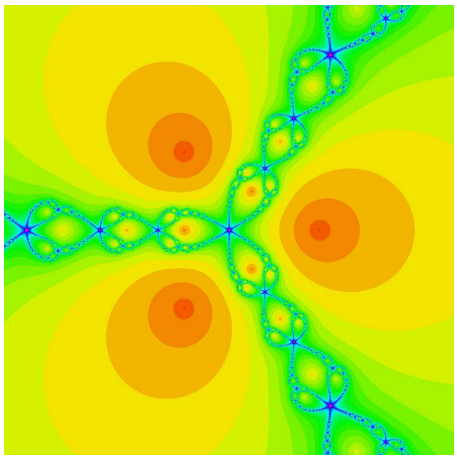
Slika 49: Newtonova metoda za iskanje ničel polinoma

Piksle na sliki asociiramo s kompleksno ravnino tako, da ravnamo z x koordinato slike kot realni del kompleksnega števila, z y koordinato pa kot imaginarni del kompleksnega števila. Ta števila uporabimo kot začetne točke Newtonove metode in glede na izbran način barvanja fraktala le-tega tudi pobarvamo. V aplikaciji sem se odločila barvati glede na število iteracij, ki jih točka potrebuje, da skonvergira k ničli (oz. zelo majhne vrednosti okoli ničle delta).

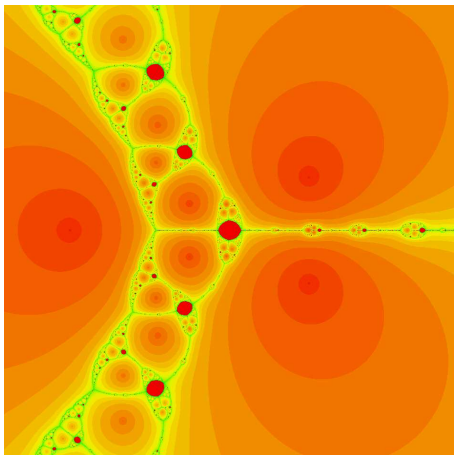
Algoritem 7 Risanje Newtonovih fraktalov

```
1: procedure RIŠI(steviloIteracij)
2:     for vsak (x, y) do
3:         (re, im) = mapiraj_koordinati_v_kompleksno_ravnino(x, y)
4:         for k = 0 do steviloIteracij do
5:             (re, im) = (re, im)^2 + f((re, im))/f'((re, im))
6:             if abs(f(re, im)) < delta then break
7:         barva = pridobi_barvo(k)
8:         pobarvaj_piksel(x, y, barva)
```

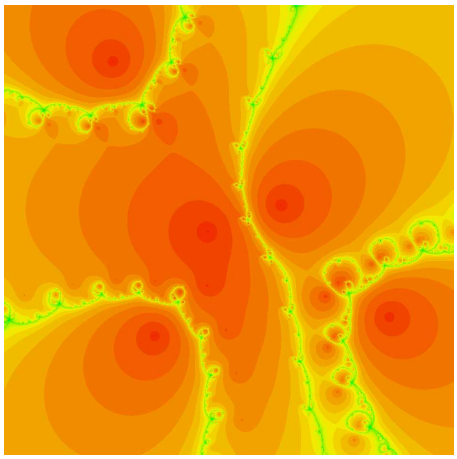
Glede na izbrane koeficiente uporabljenega polinoma dobimo zelo različne fraktale. Nekaj jih je predstavljenih na naslednjih slikah skupaj s koeficienti, s katerimi sem jih pridobila.



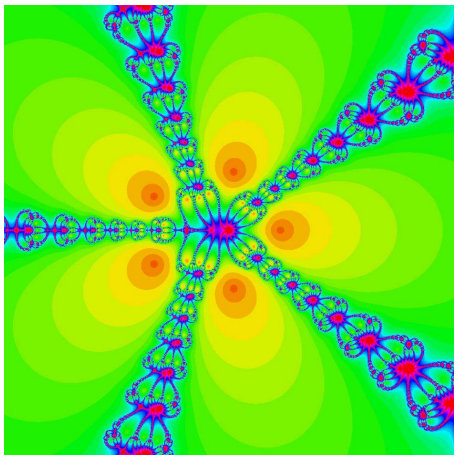
Slika 50: Newtonov fraktal za polinom $z^3 - 1$



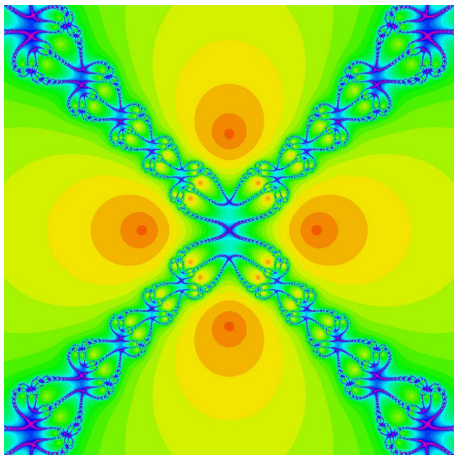
Slika 51: Newtonov fraktal za polinom $z^3 - 2z + 2$



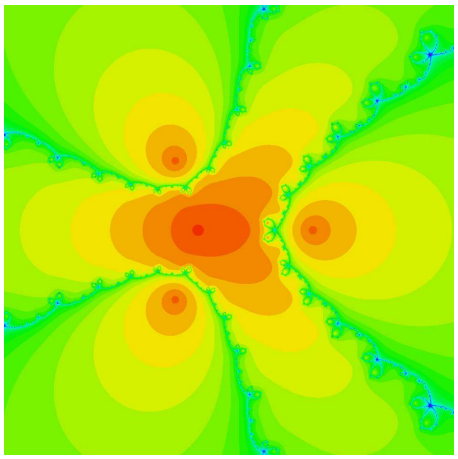
Slika 52: Newtonov fraktal za polinom $z^5 - 3iz^3 - (5+2i)z^2 + 3z + 1$



Slika 53: Newtonov fraktal za polinom $5z^5 + 5z^4 + z^3 - 1$



Slika 54: Newtonov fraktal za polinom $15z^4 + z^3 + 3z^2 - 16$



Slika 55: Newtonov fraktal za polinom $-16z^4 - 10z^3 - z^2 + 15z + 5$

5.4 Naključni fraktali

Naključni fraktali so generirani s pomočjo stohastičnih in nedeterminističnih procesov. Večinoma bo vsako generiranje naključnega fraktala podalo drugačno sliko, vendar bo slika podobna, razen, če se posebej potrudimo, da naš generator naključnih vrednosti, ki jih fraktal potrebuje za svoj razvoj, generira isto zaporedje.

5.4.1 Brownovo drevo

Brownovo drevo je dobilo svoje ime po Brownovem gibanju, ki opisuje naključno gibanje delcev v tekočini ali plinu, nastalo zaradi trkov s hitrimi atomi ali molekulami tekočine ali plina.

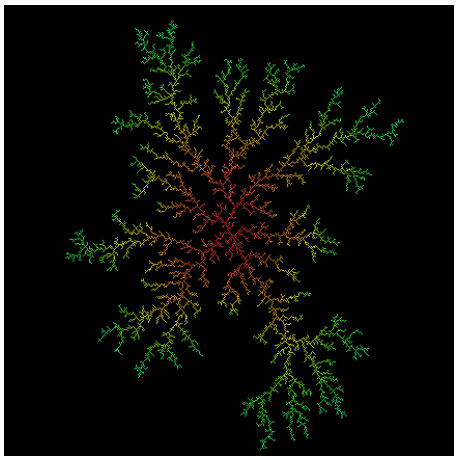
Brownova drevesa so matematični modeli dendritnih struktur in opisujejo difuzijsko omejeno agregacijo (Diffusion limited aggregation) le-teh. Difuzijsko omejena agregacija je proces, pri katerem gibajoči se delci v Brownovem gibanju kot posledica naključnih sprehodov, agregirajo v skupke. To lahko opazimo tudi v realnem svetu pri elektrolizi, mineralnih depozitih in drugje.

Brownova drevesa so grajena s pomočjo naslednjih korakov: najprej postavimo na sliko seme. Okoli tega semena se kasneje začne graditi naš depozit. Na naključno pozicijo na sliki postavimo delec, ki ga naključno premikamo, dokler se ne zaleti v seme oz. že nastali depozit. Delec pustimo oz. narišemo na tem mestu, kjer se je zaletel in postavimo nov delec na sliko. Ta postopek ponavljamo. Začetno pozicijo delcev sem postavila v krog okoli semena na sredini slike. Velikost strukture sem omejila tako, da ni presegla meje kroga, iz katerega sem pošiljala delce.

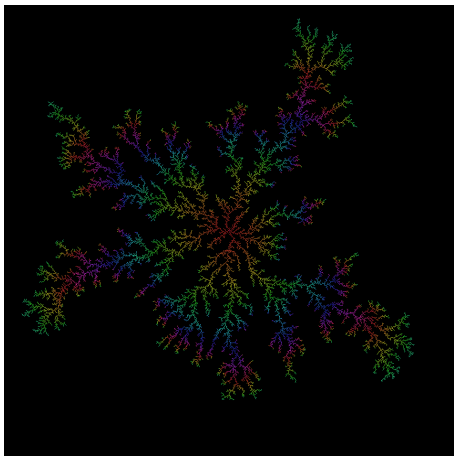
Algoritem 8 Risanje Brownovega drevesa

```
1: procedure NAKLJUČNI SPREHOD DELCEV(visinaSlike)
2:     maxPolmer = visinaSlike/2
3:     polmer = 0
4:     postavi_seme_na_sredino_slike()
5:     while polmer ≤ maxPolmer do
6:         (x, y) = naključna_pozicija_na_krogu_okoli_semena()
7:         while true do
8:             smer = pridobi_nakljucno_smer_premika()
9:             (x, y) = premakni_delec(smer)
10:            if delec_izven_kroga(x, y) then break
11:            if katerokoli_sosednje_polje_zasedeno(x, y) then break
12:            plot(x, y)
13:            if razdalja((x, y), (x_seme, y_seme)) ≥ polmer then
14:                polmer = polmer + 1
```

Nastalo drevo ima lahko veliko različnih oblik, odvisnih od pozicije semena, začetne pozicije delcev in algoritma naključnega premikanja. Na sliki je Brownovo drevo, katerega nastanek sem v aplikaciji tudi animirala.



Slika 56: Brownovo drevo



Slika 57: Brownovo drevo 1

5.4.2 Fraktalna pokrajina

Fraktalna pokrajina je generirana s stohastičnim algoritmom, ki imitira izgled naravne pokrajine. Ta algoritem ni deterministična fraktalna površina, ampak generira naključno površino, ki ima fraktalne lastnosti.

Če si gore ogledujemo od blizu, imajo navidezno enake lastnosti, kot če si jih ogledujemo od daleč. To velja tudi za zelo majhne kamene strukture, ki imajo prav tako podobne lastnosti. Na podlagi tega principa se generirajo fraktalne pokrajine.

Začnemo z naključo površino in rekurzivno dodajamo naključne podrobnosti, ki oponašajo strukturo celote, vendar na vedno manjših površinah. Uporabila sem algoritem Dragulj - kvadrat, sprva opisan leta 1982 s strani znanstvenikov Fournier, Fussel in Carpenter.

Postopek grajenja naključnega višinskega 'zemljevida' poteka po naslednjih korakih:

1. Štirim kotnim točkam mreže, ki predstavljajo naš zemljevid, določimo višino.

2. Vzamemo povprečno vrednost teh kotnih točk, dodamo naključno motnjo in to vrednost pripišemo točki, ki leži na sami sredini kvadrata. Ta korak je korak dragulja, ker s tem postopkom ustvarimo strukturo podobno diamantu, kar bi lahko videli, če bi točke grafično povezali.

3. Pri vsakem nastalem dragulju vzamemo vse štiri vogalne točke, jih povprečimo, dodamo naključno motnjo in to vrednost pripišemo točki, ki leži na sredini tega dragulja. Temu koraku rečemo korak kvadrata, saj s tem ustvarjamo nove kvadratne oblike.

4. Zadnji korak je ponavljanje točke 2. in 3., pri čemer se izmenjujeta korak dragulja in kvadrata, le da zmanjšamo polmer kvadrata in dragulja za polovico ob vsaki ponovitvi.

Naključna motnja je določena s koeficientom, ki je med 0.0 in 1.0. Ko dodajamo manjše podrobnosti, zmanjšamo tudi naključne motnje, ki jih dodajamo. Majhne podrobnosti na manjši površini so namreč fraktalno podobne velikim spremembam na večji površini.

Algoritem 9 Generiranje fraktalne pokrajine

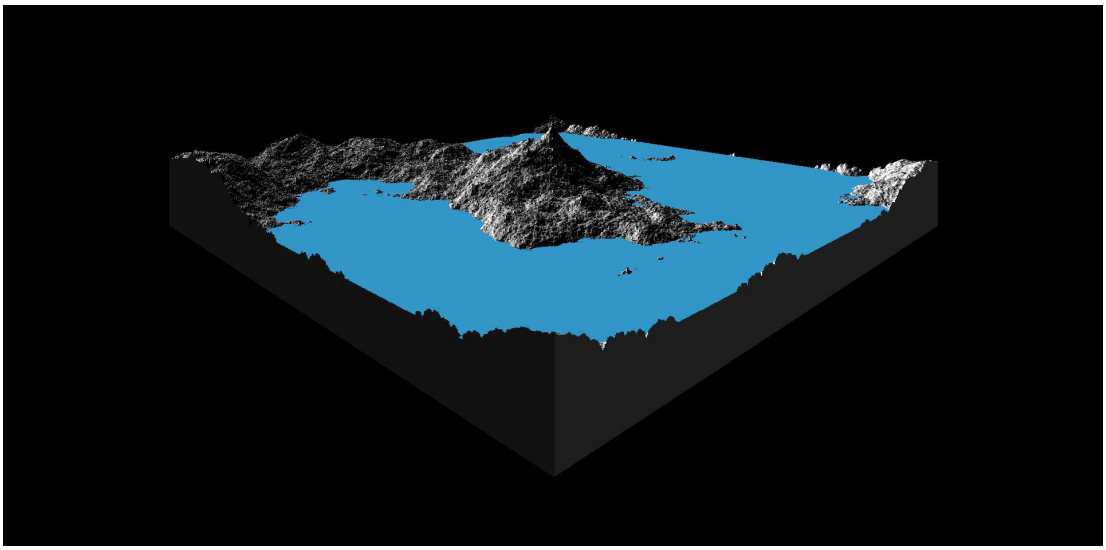
1: **procedure** GENERIRAJVISINSKOMAPO($zrnatost$, $resolucija$)
2: $velikost = 2\hat{}resolucija$
3: $maxVelikost = velikost$
4: inicializiraj_mapo_na_polje_nicel($velikost, velikost$)
5: nastavi_vogalne_vrednosti_na_mapi($mapa, velikost/2, zrnatost$)
6: deli_mapo()
7: **procedure** DELIMAPO($velikost$, $maxVelikost$, $zrnatost$)
8: $polovica = velikost/2$
9: **if** $polovica < 1$ **then** break
$povecava = zrnatost * velikost$
10: **for** $y = polovica; y < maxVelikost; y+ = velikost$ **do**
11: **for** $x = polovica; x < maxVelikost; x+ = velikost$ **do**
12: kvadrat($y, x, polovica$, nakljucno_stevilo(0, 1) * $povecava$ * (2 − $povecava$))
13: **for** $y = 0; y < maxVelikost; y+ = polovica$ **do**
14: **for** $x = y + polovica; x < maxVelikost; x+ = velikost$ **do**
15: dragulj($y, x, polovica$, nakljucno_stevilo(0, 1) * $povecava$ * (2 − $povecava$))
16: deli_mapo($velikost/2$, $maxVelikost$, $zrnatost$)
17: **procedure** KVADRAT($y, x, polovica, nakljucnaVrednost$)
18: $sredisce = (y, x)$
19: $polmer = polovica$
20: $povprecnaVrednost =$ povpreci_vrednosti_na_kvadratu($sredisce, polmer$)
21: $novaVrednost = povprecnaVrednost + nakljucnaVrednost$
22: nastavi_vrednost_na_mapi(y, x, $novaVrednost$)
23: **procedure** DRAGULJ($x, y, polovica, nakljucnaVrednost$)
24: $sredisce = (y, x)$
25: $polmer = polovica$
26: $povprecnaVrednost =$ povpreci_vrednosti_na_dragulju($sredisce, polmer$)
27: $novaVrednost = povprecnaVrednost + nakljucnaVrednost$
28: nastavi_vrednost_na_mapi(y, x, $novaVrednost$)

Ko imamo tako našo površino definirano na kvadratni mapi, moramo to tudi narisati na ekran tako, da bo izgledalo podobno realni pokrajini. To naredimo

tako, da aproksimiramo površino s kopico povezanih kvadratov, ki imajo vsak svojo barvo oz. senco.

Najprej kvadrat rotiramo za 45 stopinj, nato njegovo 3D pozicijo projeciramo v 2D prostor z matriko za perspektivno projekcijo s koeficienti, ki so določeni eksperimentalno.

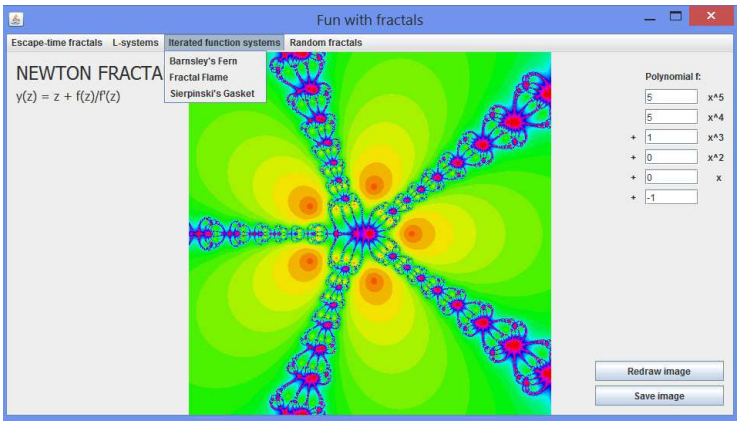
Slika 58: Fraktalna pokrajina



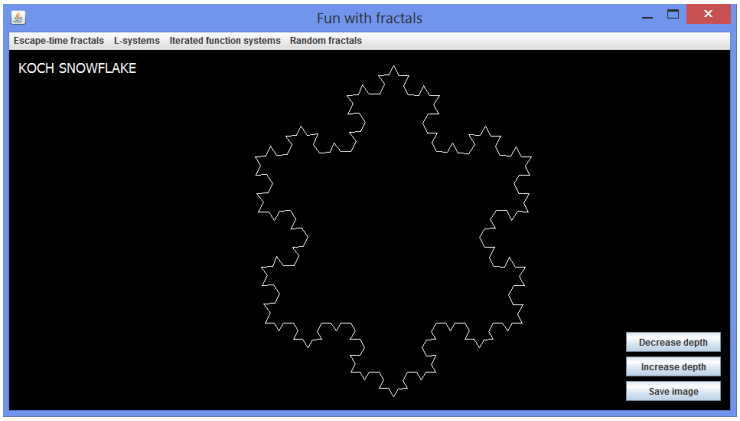
6 Rezultati

Nastala je aplikacija v angleškem jeziku, ki zna generirati vse tipe fraktalov, opisane v prejšnjem poglavju. Poimenovala sem jo 'Fun with fractals' (Zabava s fraktali). Vse slike, ki jih generira je možno shraniti.
Koda praktičnega dela diplomskega dela se nahaja na naslovu:
`https://github.com/xtrinch/fractal_generator`.

Slika 59: Zaslonska slika aplikacije 1



Slika 60: Zaslonska slika aplikacije 2



7 Zaključek

Upam, da bo komu moje delo, raziskovanje in aplikacija pripomogla, da se bo bolj povezal z matematiko, umetnostjo in z naravo/naravnimi oblikami. Po raziskavi področja kot je ta, nikoli več ne gledaš na zamrznjeno okno z ledenimi strukturami, oblake, cvetačo ali pokrajino na enaki način.

Pričakujem, da bo na področju fraktalov odkrito še veliko zanimivega in da bodo znanstveniki našli vedno več vzporednic z današnjim življenjem in modeliranjem naravnih sistemov.

Literatura

[1] Drawing fractals with interval arithmetics. `http://www.rawnsley.info/rupert/research/interval/index.htm`. Datum dostopa: 12.23.2014.

[2] Fraktalne strukture. `http://revija.fmf.si/2009/11/fraktalne-strukture/`. Datum dostopa: 12.23.2014.

[3] Java applet programming. `http://java.rubikscube.info/`. Datum dostopa: 12.23.2014.

[4] M. F. Barnsley and S. Demko. Iterated Function Systems and the Global Construction of Fractals. *Proceedings of The Royal Society A: Mathematical, Physical and Engineering Sciences*, 399:243–275, 1985.

[5] Kenneth Falconer. *Fractal geometry: mathematical foundations and applications*. John Wiley & Sons, 2013.

[6] Kenneth J Falconer and KJ Falconer. *Techniques in fractal geometry*, volume 16. Wiley Chichester (W. Sx.), 1997.

[7] Benoit B Mandelbrot. *Fractals: form, chance and dimension*. Freeman, San Francisco, CA, 1977.

[8] Heinz-Otto Peitgen and Dietmar Saupe, editors. *The Science of Fractal Images*. Springer-Verlag New York, Inc., New York, NY, USA, 1988.

[9] Daniel Shiffman. *The nature of code:[simulating natural systems with processing]*. Selbstverl., 2012.

[10] Roger T. Stevens. *Advanced Fractal Programming in C - with Disk*. IDG Books Worldwide, Inc., Foster City, CA, USA, 1990.